Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 85 (2016) 588 – 597

Procedia Computer Science

International Conference on Computational Modeling and Security (CMS 2016)

# An Approach to Reduce the Computational Burden of Nearest Neighbor Classifier

R. Raja Kumar[a,], P. Viswanath[b], C.Shobha Bindu[c]

[a] *Rajeev Gandhi College of Engg. and Technology, Nandyal. Email: rajsri1229@yahoo.co.in*

[b] *IIITS,Chittoor. Email:viswanath.p@ieee.org*

[c] *JNTUA College of Engineering, Anantapuramu. Email: shobabindu@gmail.com*

**Abstract**

Nearest Neighbor Classifiers demand high computational resources i.e, time and memory. Reducing of reference set(training set) and feature selection are two different approaches to this problem. This paper presents a method to reduce the training set both in cardinality and dimensionality in cascade. The experiments are done on several bench mark datasets and the results obtained are satisfactory.



Peer-review under responsibility of the Organizing Committee of CMS 2016

*Keywords:* Nearest Neighbour Classifiers, Editing, Feature selection, Linear Discriminant Analysis.

## 1. Introduction

Nearest Neighbor classification(NNC) is a popularly known classification method in pat-tern recognition. NNC is simple to use and easy to understand. For a test pattern q, NNC or 1-NNC finds the nearest neighbor in the reference set and assigns the class label of this nearest neighbor to the test pattern [1].

A simple enhancement of NNC is to consider the k nearest neighbors of the test pattern and the class label infers by majority voting. NNC suffers some drawbacks. 1) NNC re-quires memory space to store all the training patterns. To classify the given query pattern q, it computes the distances between the query pattern q and every training pattern from the training set in order to find the nearest neighbor of q. Hence the time complexity is O(n) where n is the training set size. To classify the test set of size m, the time complexity be-comes O(mn).This will become severe overhead if the size of n is large. One can reduce the memory and computational needs of NNC, if the size of training set is reduced considerably. Considerable number of papers were published on reducing the size of training set and hence reducing the memory and computaional demands of NNC.

Similarly, the dimensionality is also a problem if the data sets are represented by high dimensionality [2] as it leads to curse of dimensionality.



Peer-review under responsibility of the Organizing Committee of CMS 2016

doi:10.1016/j.procs.2016.05.225

The rest of the paper is organized as follows: Section 2 discuss about review of related works. section 3 discuss about algorithms and notations. In section 4 data sets are discussed briefly. Experimental results have been presented in Section 5. Conclusion is present in Section 6.

## 2. Related Work

The early method presented in this context of reducing the training set size was the condensed nearest neighbor (CNN) proposed by Hart [3]. Let D be the condensed set found by applying CNN method over the original training set D. D is called condensed iff it classifies all the training patterns from D. Then D is used instead of D to classify the test set. CNN has two disadvantages: 1) Retention of unnecessary samples 2) Occasional retention of internal rather than boundary patterns [4]. In 1976, Ivan Tomek proposed two modifications to CNN which overcome the above disadvantages of CNN [4].

In 1972, Swonger proposed an iterative condensed algorithm(ICA) [5] which allows ad-ditions and deletions from the condensed set. In 2002, Devi and Murthy proposed Modified Condensed Nearest Neighbor(MCNN) method which is a prototype selection method [6]. In this method, prototype set was built incrementally.

In 2005, Viswanath, Murthy and Bhatnagar proposed an efficient NN classifier, called overlap pattern nearest neighbor (OLP-NNC) which is based on overlap pattern graph(OLP-graph) [7]. In this paper, they built OLP-graph incrementally by scanning the training set only once and OLP-NNC works with OLP-graph. In 2007, Suresh babu and Viswanath [8] proposed the generalized weighted k-nearest Leader Classifier method which finds the relative importance of prototypes called weighting of prototypes and this weight is used in classifica-tion process.

In 2011, Viswanath and Sarma [9] proposed an improvement to the k-Nearest Neighbor Mean Classifier(k-NNMC), finds k nearest neighbors class-wise. Classification process is car-ried out with these mean patterns. In 2014, Raj kumar, Viswanath and Bindu [10] proposed a new prototype selection method for nearest neighbor classification called Other Class Nearest Neighbors(OCNN) which is based on the retaining samples that are near to decision boundary and which are crucial in the classification task.

Considerable number of papers were published on reducing the cardinality of training set but Less number of papers were published on reducing the training set size in both directions i.e., cardinality and dimensionality. In 1999, Kuncheva and Jain proposed genetic algorithm based method for simultaneous editing and feature selection [11]. In 2001, Kuncheva ex-tended the paper [11] for reducing the computational demands of nearest neighbor classifier [12]. In this paper Incremenatal Hill Climbing(IHC), and Stochastic Hill Climbing(SHC) methods were used to reduce the training set in both directions. In 2013, TR Babu, MN Murthy and SV subrahmanya published book on Data Compression Schemes for Mining Large Datasets [13].

## 3. Algorithms and Notations

Let TS be the given training set. A pattern in TS is represented as $(X_i,Y_i)$ for i=1,2,. . . n (n is the cardinality of TS) where $X_i$ represents the d-dimensional feature vector and $Y_i$ represents the class label for $i^{th}$ pattern. So, $X_i=(x_{i1},x_{i2},. . . x_{id})$ and $Y_i=\{y_1,y_2,. . . y_c\}$ where c is the no of classes present.

### *3.1. Algorithm: CNN*

Condensed Nearest Neighbor begins with the pattern selected from the training set which forms the initial condensed set. Then each pattern in the training set is classified using the condensed set. If a pattern in the training set is misclassified, it is included in the condensed set. After one pass through the training set, another iteration is carried out. This iterative process is carried out till there are no misclassified patterns. The condensed set has two properties.

- It is a subset of original training set
- It ensures 100% accuracy over the training set which is the source for deriving the condensed set.

Let TS be the given training set and TS(CNN) is the new training set formed by applying condensed nearest neighbor rule.The algorithm is outlined as shown in algorithm 1.

**Algorithm 1:** Condensed Nearest Neighbor algorithm

**1** Initially TS(CNN)=∅.
**2** TS(CNN)= select the first sample TS.
**3** Classify TS using TS(CNN).
**4 for** each sample in TS **do**
5 a) if it is correctly classified then ignore it.
6 b) if it is incorrectly classified, then add it to TS(CNN)
**7** Repeat step 3-6 till there are no new samples added to TS(CNN) *i.e.,*
**8** $TS(CNN)_i = TS(CNN)_{i-1}$ where i is the iteration number.
**9** Output TS(CNN) as condensed set over TS.

### *3.2. Algorithm: OCNN*

This method is based on the intutive notion of retaining patterns that are near to decision boundary. It is obvious that the patterns which are near decision boundary plays very impor-tant role in the classification of a process. The boundary patterns are computed by using Other Class Nearest Neighbors(OCNN) method.

Let TS be the given training set. TS(OCNN) is the set formed after applying the OCNN method. OCNN($X_i$) is the set of nearest neighbors for pattern $X_i$ from other classes which can be computed by using k-nn rule. The method is outlined in algorithm 2.

**Algorithm 2:** Other Class Nearest Neighbor algorithm

**1** Initially TS(CNN)=∅.
**2** TS(CNN)= select the first sample TS.
**3** Classify TS using TS(CNN).
**4 for** each sample in TS **do**
5 a) if it is correctly classified then ignore it.
6 b) if it is incorrectly classified, then add it to TS(CNN)
**7** Repeat step 3-6 till there are no new samples added to TS(CNN) *i.e.,*
**8** Output TS(CNN) as condensed set over TS.

Consider the example training set shown in Figure 1. Two classes are present in the Figure 1. The positive class tuples are represented by '+' and negative class tuples are represented by '-'. For example, if a query pattern '*' lies on the left side of the boundary decision, then it is classified as '+' class otherwise '-' class. That means the patterns that are near to decision boundary are enough for classification. The other samples can be removed from the training set. This is the central idea of our method. In the example present in the Figure 1,two patterns from '-' class and two patterns from '+' class near to decision boundary are typical patterns when compared to other training patterns. This can be found by using the Other Class Nearest Neighbors algorithm(OCNN) discussed in algorithm 2.

### *3.3. Linear Discriminant Analysis(LDA)*

LDA is a Dimensionality reduction technique used in machine learning. It is also one of the preprocessing technique. LDA extracts the good features and project the dataset onto a lower dimensional space with good-class separability. It is similar to Principal Component Analysis but in addition to finding the component axes that maximize the variance of data, we are additionally interested in the axes that maximize the seperation between the classes. The LDA process is carried out as follows [14].

1) Compute the mean vectors for the different classes from the dataset.

2) Compute the in-between-class scatter matrix $S_b$ and within-class-scatter matrix $S_w$. $S_w$ can be computed as

$$S_w = \sum_{i=1}^{c} S_i \tag{1}$$

where i=1,2...c *i.e* c is number of classes. and $S_i$ is is scatter matrix of class i

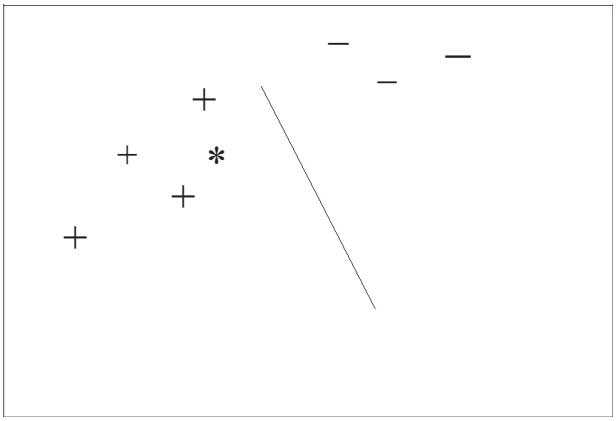


Figure 1. Nearest Neighbor Classifier

$$S_i = \sum_{xD_i} (x - m_i)(x - m_i)^t \tag{2}$$

where $m_i$ is mean of class i.

$$m_i = \frac{1}{n} \sum_{xQD_i} x \tag{3}$$

where $D_i$ is dataset of class i. The in-between-class scatter matrices can be computed as

$$S_b = \sum_{i=1}^{c} (m_i - m)(m_i - m)^t . \tag{4}$$

The objective function is

$$J(V) = \frac{det(V^t S_b V)}{det(V^t S_w V)} \tag{5}$$

where V is the projection matrix to the subspace of dimensionality k=c-1 atmost( c is the number of classes). This can be converted to general eigen value problem.

$$S_b V = \lambda S_w V \tag{6}$$

$$\lambda V = S_w^{-1} S_b V \tag{7}$$

Let $V_1, V_2 ... V_{c-1}$ be the corresponding eigen vectors. The projection matrix V is of order d × k used to project the data onto subspace of dimension k(k ≤ d) is given by the eigen vectors corresponding to the largest k eigen values. V projects the dataset to subspace atmost c-1
where c is no of classes using the following computation.

$$Y = V^t X \tag{8}$$

where Y is projected pattern of order k × 1 and X is of order d × 1.

*3.4. An Example:*

We illustrate the algorithms with the example dataset shown in table 1. It has two at-tributes X and Y along with class label present in the last column. Two class labels Yes and No are present. Let $X_i$ refers to the pattern i in the dataset *i.e*,$X_1$ is the first pattern (1,1) in the dataset. The euclidean distances between the patterns are present in the table 2.

1) Now, we build prototype set S based on the method OCNN shown in algorithm 2.

Table 1. Example dataset

| No. | A | B | Class |
|---|---|---|---|
| $X_1$ | 1 | 1 | Yes |
| $X_2$ | 1 | 2 | Yes |
| $X_3$ | 2 | 1 | Yes |
| $X_4$ | 2 | 2 | Yes |
| $X_5$ | 5 | 1 | No |
| $X_6$ | 5 | 2 | No |
| $X_7$ | 6 | 1 | No |
| $X_8$ | 6 | 2 | No |

The $S_{Xi}$ =OCNN($X_i$) where $i$ = 1,2,...8 obatained as follows. we considered k=2.

$S_{X1}$ =OCNN($X_1$)={$X_5$,$X_6$} $S_{X2}$ =OCNN($X_2$)={$X_5$,$X_6$}
$S_{X3}$ =OCNN($X_3$)={$X_5$,$X_6$} $S_{X4}$ =OCNN($X_4$)={$X_5$,$X_6$}
$S_{X5}$ =OCNN($X_5$)={$X_4$,$X_3$} $S_{X6}$ =OCNN($X_6$)={$X_4$,$X_3$}
$S_{X7}$ =OCNN($X_7$)={$X_4$,$X_3$} $S_{X8}$ =OCNN($X_8$)={$X_4$,$X_3$}

The final prototype set S is computed by the union of all $S_{Xi}$ . For this example, S=$S_{X1} \cup S_{X2} \cup S_{X3} \cup S_{X4} \cup S_{X5} \cup S_{X6} \cup S_{X7} \cup S_{X8}$ . Hence, we get S={$X_5$,$X_6$,$X_4$,$X_3$} shown in table 3.

2) We build the prototype set S based on the method CNN shown in algorithm 1.
The first pattern $X_1$ forms the initial condensed set S. So S={$X_1$}. Using S, the dataset in table 1

Table 2. Euclidean distances between patterns of table 1

| No. | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ |
|---|---|---|---|---|---|---|---|---|
| $X_1$ | 0 | 1 | 1 | 1.414 | 4 | 4.12 | 5 | 5.09 |
| $X_2$ | 1 | 0 | 1.414 | 1 | 4.12 | 4 | 5.09 | 5 |
| $X_3$ | 1 | 1.414 | 0 | 1 | 3 | 3.16 | 4 | 4.12 |
| $X_4$ | 1.414 | 1 | 1 | 0 | 3.16 | 3 | 4.12 | 4 |
| $X_5$ | 4 | 4.12 | 3 | 3.16 | 0 | 1 | 1 | 1.414 |
| $X_6$ | 4.12 | 4 | 3.16 | 3 | 1 | 0 | 1.414 | 1 |
| $X_7$ | 5 | 5.09 | 4 | 4.12 | 1 | 1.414 | 0 | 1 |
| $X_8$ | 5.09 | 5 | 4.12 | 4 | 1.414 | 1 | 1 | 0 |

Table 3. Prototype set choosen by OCNN on table 1

| No. | A | B | Class |
|---|---|---|---|
| $X_3$ | 2 | 1 | Yes |
| $X_4$ | 2 | 2 | Yes |
| $X_5$ | 5 | 1 | No |
| $X_6$ | 5 | 2 | No |

is classified. During the first iteration, the patterns $X_1$,$X_2$,$X_3$,$X_4$ are ignored as they classify using S. The pattern $X_5$ is added to S because it is miscalssified by S. The patterns $X_6$,$X_7$,$X_8$ are ignored as they classify using S. After the end of the first iteration, S={$X_1$,$X_5$}. Then the second iteration is carreid out. During the second iteration new patterns are not added to S because all the patterns in the dataset are classified. At this stage, the algorithm is terminated with set S={$X_1$,$X_5$}, the final condensed prototype set. In this section, the data sets used for the experiments are discussed. We applied the methods on 5 data sets. All the data sets are taken from Murphy and Aha [15]. The information about the data sets is shown in table 4. It shows about the name of the data sets, number of training and testing patterns, no of features and also no of classes.

## 4. Experimental Results

We experimented the method with the popular data sets. The accuracies obtained on the data sets are shown in table 5. In this table, the accuracies obtained by OCNN and CNN are given. We got satisfactory results.

Table 4. Description of Data sets

| Data set | No.of training Patterns | No.of Test Patterns | Number of dimensions | Number of classes |
|---|---|---|---|---|
| Wine | 120 | 57 | 13 | 3 |
| Thyroid | 144 | 71 | 5 | 3 |
| Iris | 99 | 51 | 4 | 3 |
| Balance | 416 | 209 | 4 | 3 |
| Optical.digit.rec | 3823 | 1797 | 64 | 10 |

Table 5. Accuracy obtained over data sets

| | OCNN | | CNN | |
|---|---|---|---|---|
| Data Set | No of Prototypes | No of Accuracy % | Prototypes | Accuracy % |
| Wine | 45 | 94.74 | 27 | 94.74 |
| Thyroid | 38 | **92.10** | 16 | 91.55 |
| Iris | 22 | **98.33** | 16 | 96.08 |
| Balance | 214 | 74.64 | 143 | 71.77 |
| Opt.digit.rec | 622 | **96.55** | 305 | 96.38 |

These results are evident enough to say that OCNN method is competing with CNN method and it is showing good accuracy. The methods are implemented on 'DELL OPTI-PLEX 740 n' model having Intel core 2 DUO 2.2 Ghz processor with 1GB DDR 2 RAM capacity. We compared the time taken to compute the prototype set by OCNN method and by CNN method. The results are shown in the table 6. From the table, it is clear that the OCNN method is showing good improvement in execution time with respect to CNN over all datasets. OCNN method can build the prototype set faster than CNN. This is because, OCNN method does not require multiple scans as CNN.

By using OCNN and CNN prototype selection methods we can reduce the size of the data set but the dimensionality remains same. To reduce the dimensionaliy, we used LDA discussed in section 3.3 There are two options now.

1. Applying prototype selection method followed by dimensionality reduction method.
2. Applying dimensionality reduction method followed by prototype selection method.

Table 6. Comparision of Time

| Data set | OCNN | CNN |
|---|---|---|
| Wine | 0.010s | 0.019s |
| Thyroid | 0.005s | 0.009s |
| Iris | 0.004s | 0.008s |
| Balance | 0.013s | 0.022s |
| Optical.digit.rec | 3.913s | 5.001s |

We experimented with the first posssibility. The results are tabulated according to the dataset wise. OCNN+LDA means first OCNN is applied and then LDA is applied. The results over thyroid dataset are given in Table 7.

Table 7. Accuracy obtained over Thyroid

| | OCNN+LDA | | CNN+LDA | |
|---|---|---|---|---|
| k | No of Prototyes | Accuracy% | No of protoypes | Accuracy% |
| 3 | 43 | 69.73 | 24 | 19.73 |
| 5 | 43 | 65.78 | 24 | 19.73 |
| 7 | 43 | 67.10 | 24 | 67.10 |
| 9 | 43 | 67.10 | 24 | 67.10 |
| 11 | 43 | 67.10 | 24 | 67.10 |
| | | 67.36±1.28 | | 48.15±21.20 |

The graphical results obtained over thyroid dataset are present in Figure 2. The results for the iris data set are present in Table 8.
The grphical results obtained over Iris dataset are present in Figure 3.
The results obtained over Wine dataset are present in Table 9.
The graphical results obtained over wine dataset are present in Figure 4.
The results obtained over Balance Dataset are present in Table 10.

The datasets Iris,Wine,Thyroid and Balance are medium datasets. The maximum dimensionality is 13(Thyroid) and cardinality is 425(Balance). All these datasets have 3 classes so we reduced the dimensionality upto maximum 2.

To test the methods on Large datasets, we have taken large dataset, Optical digit recognition which has 64 features and dataset size is 5620. It has 10 classes. The dimensionality of the dataset is reduced to 9 still achieving considerable accuracy. The results over Optical digit recognition is present in Table 11.

Table 8. Accuracy obtained over Iris

| | OCNN+LDA | | CNN+LDA | |
|---|---|---|---|---|
| k | No of Prototyes | Accuracy% | No of protoypes | Accuracy% |
| 3 | 38 | 81.66 | 22 | 75.00 |
| 5 | 38 | 73.33 | 22 | 68.33 |
| 7 | 38 | 78.33 | 22 | 68.33 |
| 9 | 38 | 78.33 | 22 | 68.33 |
| 11 | 38 | 78.33 | 22 | 68.33 |
| | | 77.77±3.42 | | 70.55±3.14 |

Table 9. Accuracy obtained over Wine

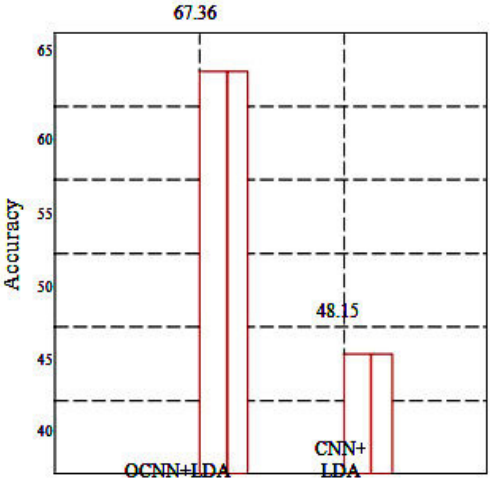
| | OCNN+LDA | | CNN+LDA | |
|---|---|---|---|---|
| k | No of Prototyes | Accuracy% | No of protoypes | Accuracy% |
| 3 | 60 | 29.31 | 22 | 25.86 |
| 5 | 60 | 25.86 | 22 | 27.56 |
| 7 | 60 | 43.10 | 22 | 25.86 |
| 9 | 60 | 44.82 | 22 | 24.13 |
| 11 | 60 | 43.33 | 22 | 25.86 |
| | | 37.28±8.01 | | 25.85±1.09 |

Table 10. Accuracy obtained over Balance

| | OCNN+LDA | | CNN+LDA | |
|---|---|---|---|---|
| k | No of Prototyes | Accuracy% | No of protoypes | Accuracy% |
| 3 | 163 | 62.85 | 143 | 60.00 |
| 5 | 163 | 63.81 | 143 | 61.90 |
| 7 | 163 | 57.14 | 143 | 60.95 |
| 9 | 163 | 58.09 | 143 | 58.57 |
| 11 | 163 | 56.19 | 143 | 56.19 |
| 13 | 163 | 57.61 | 143 | 56.19 |
| | | 59.28±2.93 | | 58.96±2.20 |

Table 11. Accuracy obtained over Opt.digit.rec

| | OCNN+LDA | | CNN+LDA | |
|---|---|---|---|---|
| k | No of Prototyes | Accuracy% | No of protoypes | Accuracy% |
| 3 | 2313 | 62.36 | 1514 | 63.14 |
| 5 | 2313 | 61.93 | 1514 | 61.83 |
| 7 | 2313 | 56.29 | 1514 | 58.09 |
| 9 | 2313 | 57.24 | 1514 | 54.52 |
| 11 | 2313 | 56.59 | 1514 | 56.88 |
| 13 | 2313 | 56.29 | 1514 | 53.21 |
| | | 58.50±2.59 | | 57.94±3.59 |



Firg:2 Results over Thyroid

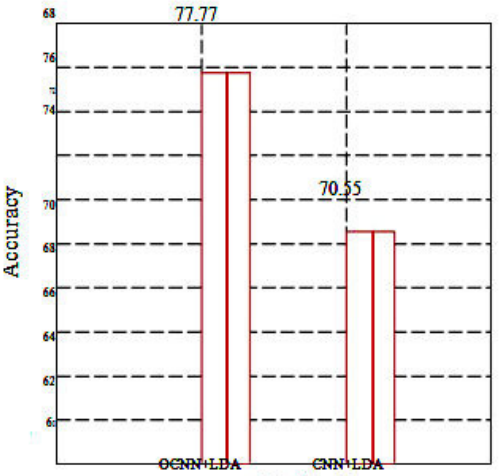


Fig:3 Results over Iris

## 5. Conclusions and Future Enhancement

In this paper, We elucidate a new method which can reduce the computational demands of the NNC. This is achieved by reducing the datasets both in dimensionality and cardinality applied in cacsade. The methods are experimented over the bench mark datasets and results are found satisfactory.

The future enhancement of the paper is, we experimented with the first possibility only and it can be extended to second possibility of reducing the computational demands of NNC *i.e.* first applying the feature selection method and then applying the prototype selection method. The work can be extended to find the answers to the questions, which one is better among the two possibilites? how to apply these methods simultaneously ? Is it possible ? If so What happens to the accuracy ?.

## References

[1] Cover T. and Hart P.: ”Nearest neighbor pattern classification”, *IEEE Transactions on Information Theory*, vol7.

[2] Verleysen, Michel, and Damien Franois.: ”The curse of dimensionality in data mining and time series pre-diction.”*Computational Intelligence and Bioinspired Systems*,Springer Berlin Heidelberg, pp 758-770, 2005.

[3] Hart P.: ”The condensed nearest neighbor rule”, *IEEE Trans on Information Theory*, vol.IT-14(3), pp 515-516, 1968.

[4] Tomek I.: ”Two modifications of cnn”, *IEEE Trans on Syst.Man.Cybern.*, vol.SMC-6 no 11, pp 769-772, 1976.

[5] Swonger, C.W.: ”Sample set condensation for a condensed nearest neighbor decision rule for pattern recog-nition”, *Front.Pattern Recognition*, pp 511-519, 1972.

[6] M.N.M and Susheela Dev, V.: ”An incremental prototype set building technique”, *Pattern Recognition.*, vol.35, pp 505-513, 2003.

[7] Viswanath, P., Murthy, N. and Bhatnagar, S.: ”Overlap pattern synthesis with an efficient nearest neighbor classifiers”, *Pattern Recognition.*, vol.38, no.8, pp 1187-1195, 2005.

[8] Babu, V.S. and Viswanath, P.: ”Weighted k-nearest leader classifier for large data sets”, *Pattern Recognition and Machine*

*Intelligence*, pp.17-24, 2007.

[9] Sarma,T.H. and Viswanath, P.: "An improvement to k-nearest neighbor classifiers", *Recent advances in Intelligent Computational Systems(RAICS)2011 IEEE*, pp 227-231, 2011.

[10] Raja kumar, R., Viswanath, P., Shoba Bindu, C.: "A New Prototye Selection Method for Nearest Neighbor Classification", *Proc. of Int. Conf. on Advances in Communication, Network, and Computing, CNC, 2014 ACEEE*, pp 1-8, 2014.

[11] Kucheva, Ludmila, I. and Jain, Lakshmi, C.: "Nearest Neighbor Classifier: Simultaneous editing and feature selection", *Pattern Recognition Letters*, vol.2, no.11, pp 1149-1156, 1999.

[12] Kuncheva, Ludmila, I.: "Reducing the Computational demand of the nearest neighbor classifier", 2001.

[13] Babu, Ravindra, T., Narasimha Murthy, M. and Subrahmanya S.V.: Data Compression Schemes for Mining Large Datasets, In *Springer London*,(2013).

[14] Sebastian Raschka 2013,http://www.sebastianraschka.com/.

[15] Murphy D.A.P.M.: "Uci repository of machine learning databases",*[http://www.ics.uci.edu/mlearn/MLRepository.html],Department of Information and Computer Science,University of California, Irvine, CA*., 1994.